Real-Time Architectures 2013/2014

Prior knowledge
Computer Systems

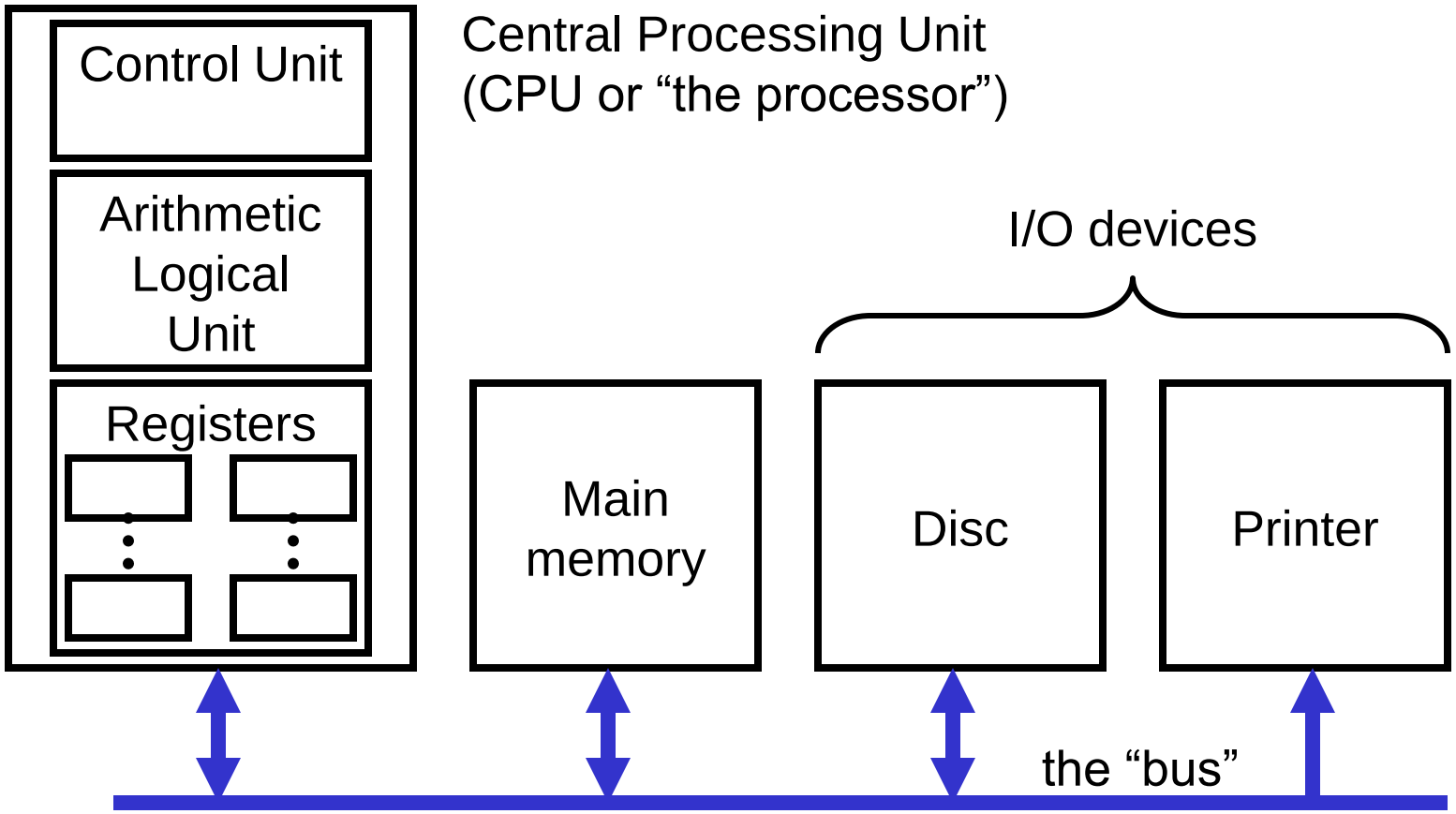
Reinder J. Bril

(By courtesy of Michael Franssen)

Contents

- Computer Systems Organization – basics
  - Architecture
  - CPU
  - Memory
  - I/O-devices
    - levels of “intelligent” control
    - interrupts
- Parallelism and cache
- Further Reading

The “architecture” of a computer

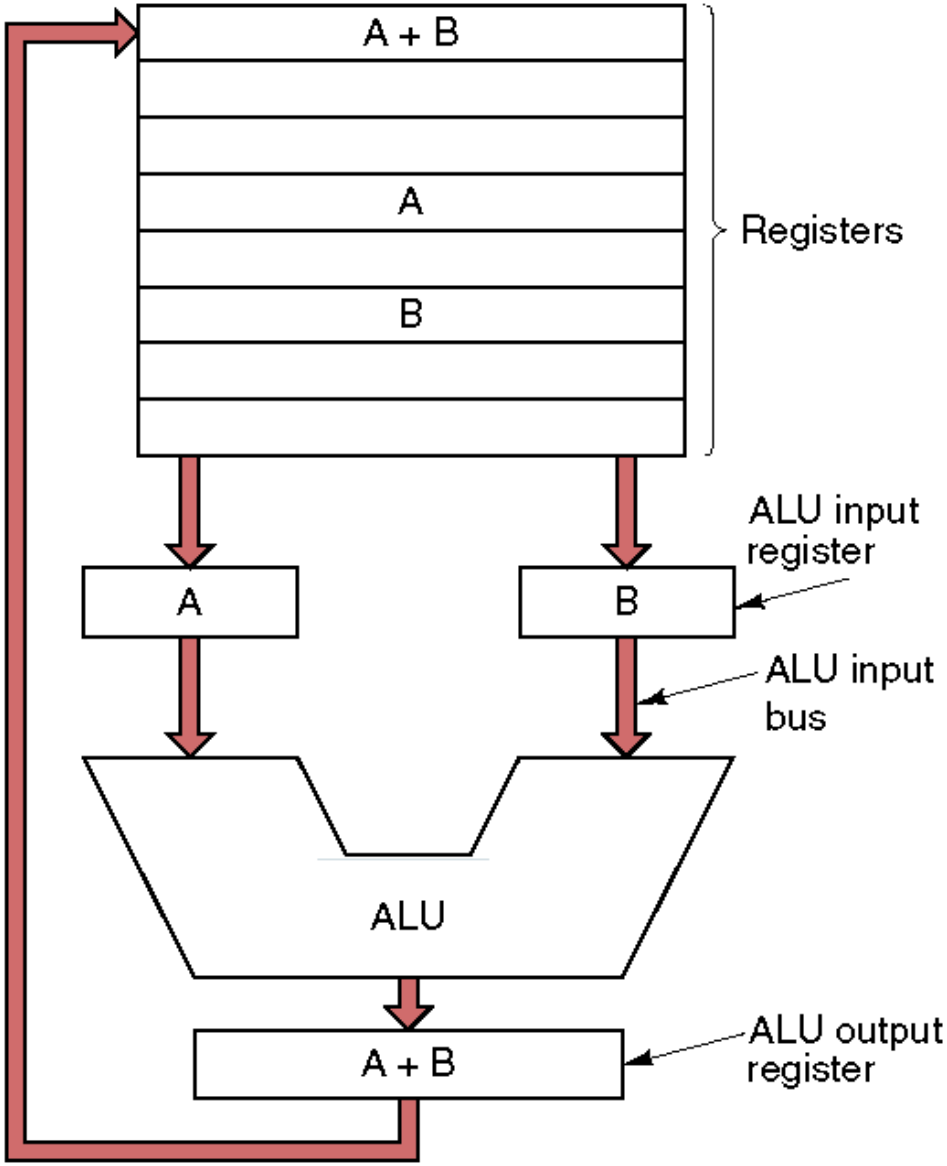


The “Central Processing Unit” (CPU)

- Control unit (CU) organizes everything, i.e.
  - fetching, decoding and executing instructions: the so-called *“fetch-decode-execute”* cycle
- Arithmetic logical unit (ALU) performs operations to carry out the instructions
- Registers are fast (and small) memory
  - “program counter” (PC): points to the next instruction to be executed
  - “instruction register” (IR): holds the instruction currently being executed
  - general/special registers: to store temporary results

The “data-path”

- Registers and ALU
- “Aux registers”
  - in hardware
  - not in HL program
- “Heart” of a computer
  - determines speed
  - determines “power”



The “fetch-decode-execute” cycle

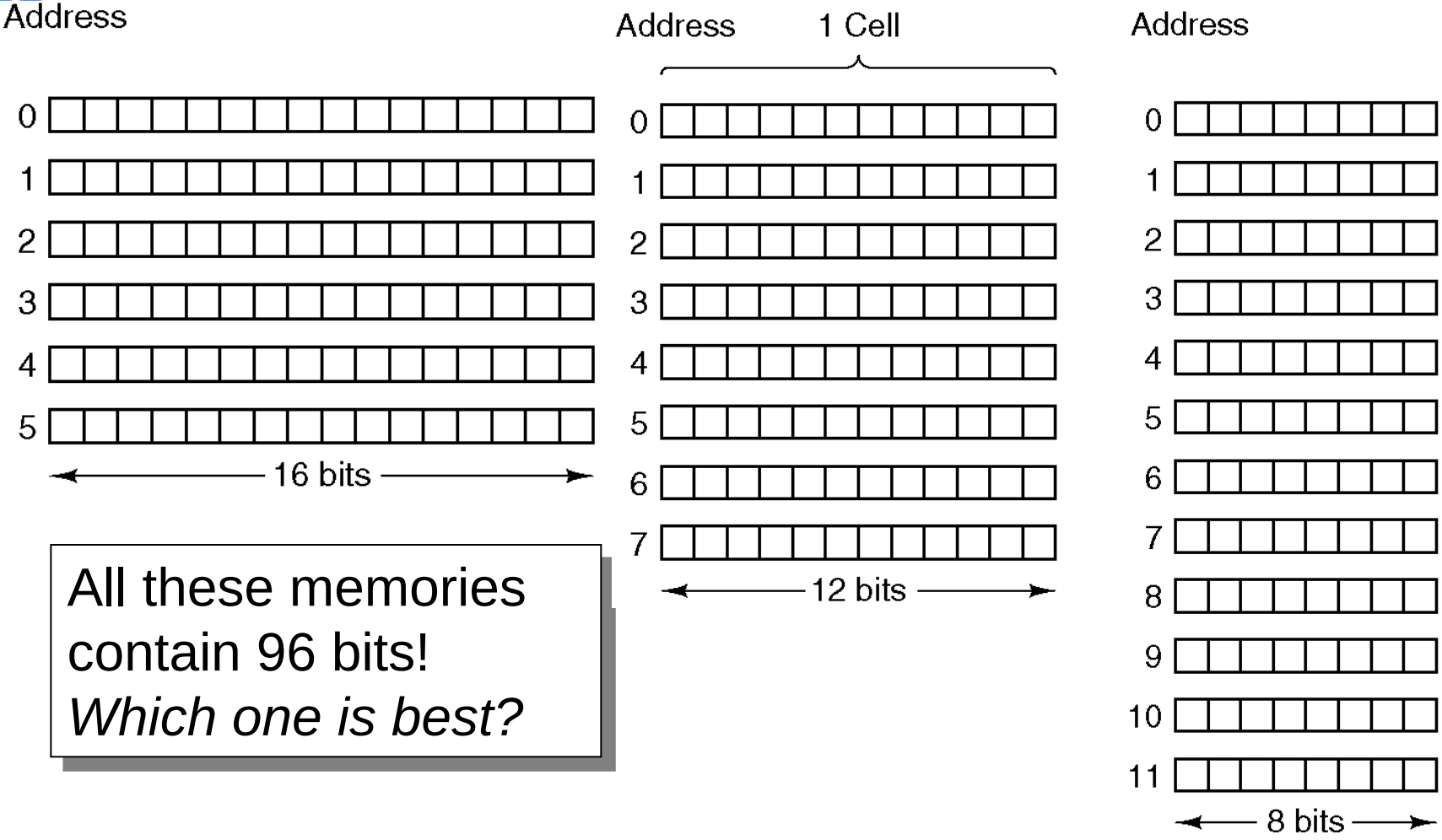
The CPU executes instructions in a series of small steps:

1. Fetch the instruction from memory (location: PC) into the IR
2. Change PC so that it points to the following instruction
3. Determine the type of instruction just fetched: register/register or register/memory
4. (Determine where the data is located in memory)
5. (Fetch the data into internal CPU registers)
6. Execute the instruction
7. Store the results at the proper place (e.g. memory)
8. Go to step 1 to begin executing the following instruction.

“Memory” contains programs and data

- Basic unit of storage: ‘Bit’ (0 or 1)
- Decimal numbers (0..9) can be stored by 4 bits
  - Inefficient: 4 bits can store 16 values!
    Calculations based on decimal numbers is no longer standard.
- Memory partitioned in “cells” (or “locations”)
  - Every cell has a fixed number of bits: $2^{(\#\text{bits in cell})}$ values
  - Every cell has its own number (called “address”), 0..(#cells - 1)
  - Address is typically a binary number: $2^{(\text{address bits})}$ cells

Three ways to organize memory



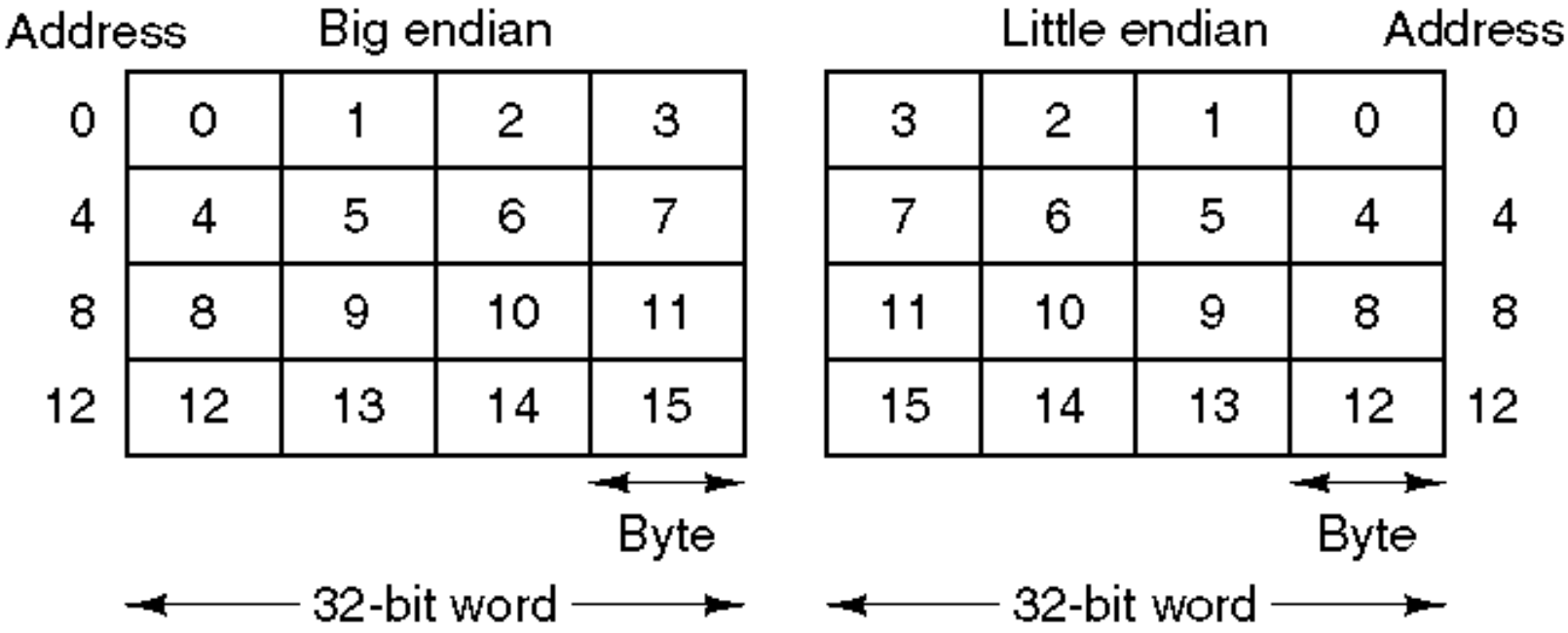
All these memories contain 96 bits!
Which one is best?

Additional organizations of memory

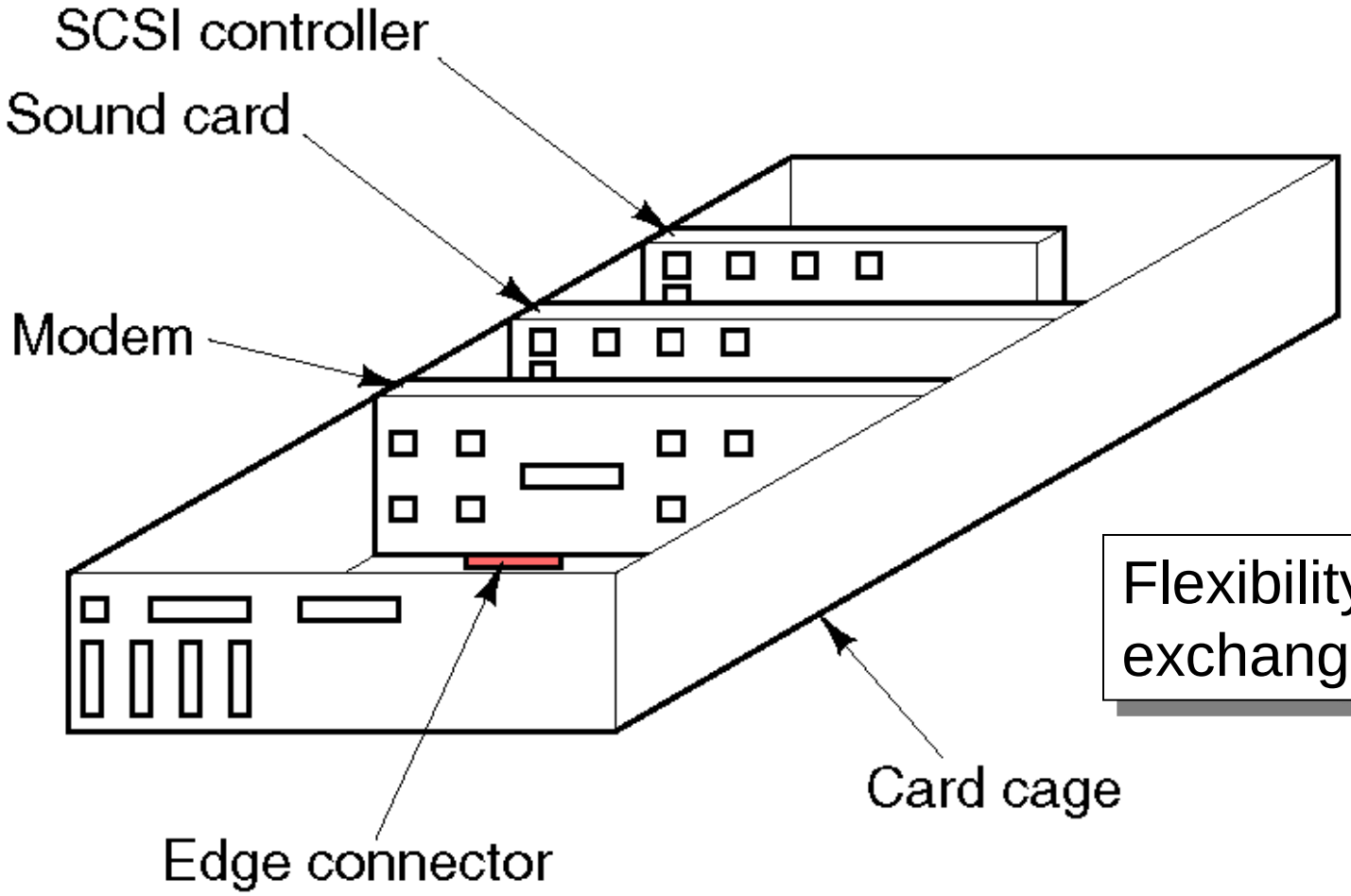
- Number of bits in a cell (now a days) typically a power of two
- 2^3 (8) is most common number of bits/cell: “byte”
  - memory can be addressed on a byte basis
- Cells are grouped in “words”
  - e.g. 2, 4 of 8 bytes per word = 16, 32 or 64 bits
  - Reading/writing words from/to memory: *faster*
  - Calculating with words : *higher precision/faster*

Order of bytes in a word

- Highest address at right end: *Big* endian
- Lowest address at right end: *Little* endian



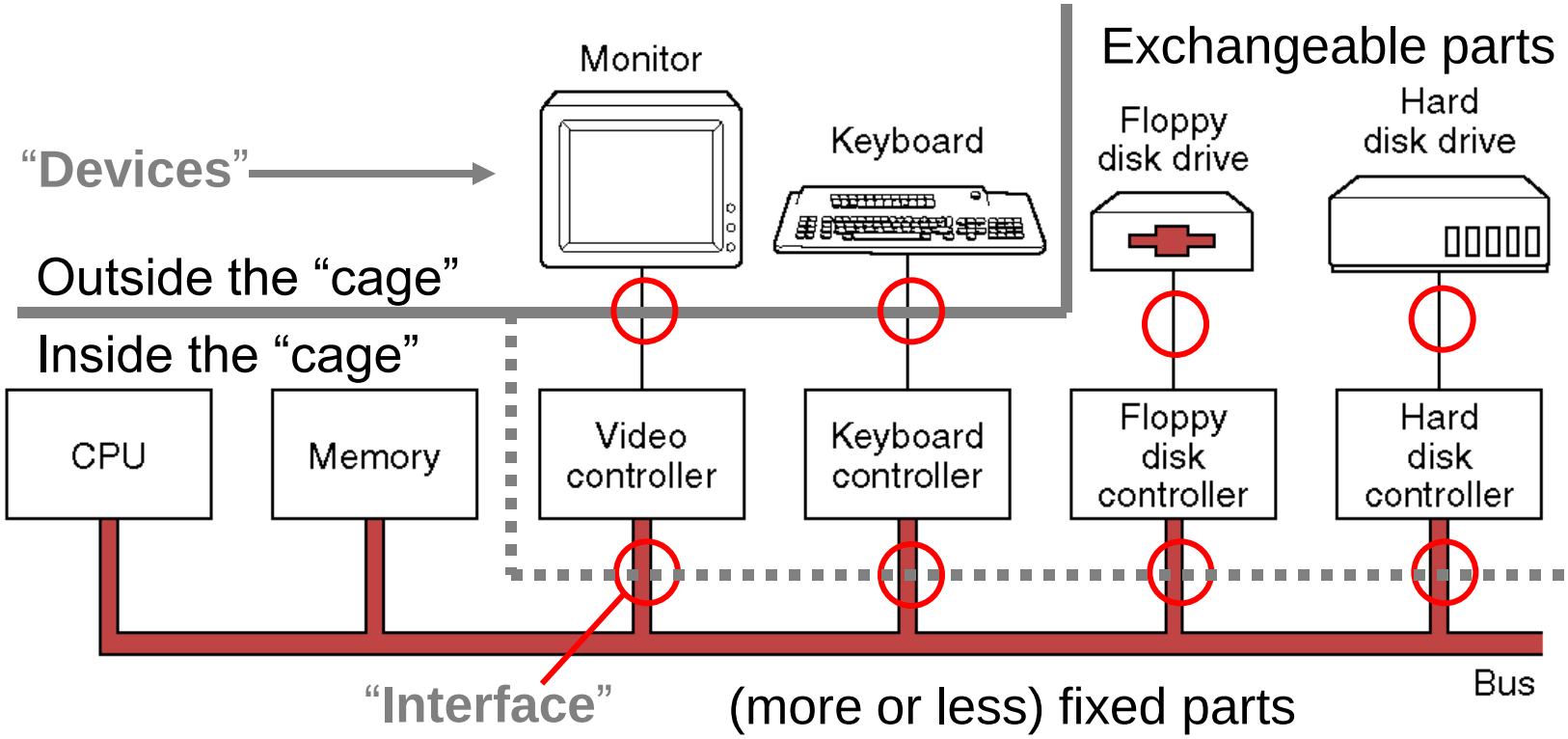
Connection with “the outside world”



Flexibility by exchangeability

“Devices” and “controllers”

- Additional flexibility by functional subdivision:



Input/output terminology

- "Device" is a physical "thing" in the outside world
  - "Outside world" = outside CPU and memory
  - Disk is "background memory"
- "Controller" connects device and CPU
  - Electrical conversion and ("intelligent") control
- "Interface" is an agreement for the connection
  - Physical (plug!), electrical and functional
  - Standardized for exchangeability

Levels of “intelligent” control

- Basic: just an electrical converter
  - Outside world mapped on memory words
  - *Disadvantage*: regular checking by CPU required
- Improved: automatic notification
  - Running program is “interrupted”
  - *Disadvantage*: support by CPU required
- Advanced: independent processing
  - Controller can read/write memory: DMA
    - “Direct Memory Access”, typically provided by dedicated controllers “supporting” the device controller
  - “Arbiter” required for memory access
  - Only a single interrupt upon completion

Input / output based on automatic notification

- The need for *interrupts* arises when input and output can proceed *in parallel* with CPU execution.
- Interrupt: activity out the outside world “breaks-in” during the execution of instructions – basics:
  - Outside world provides 1 bit to the processor
  - Processor hardware tests this bit *after* the execution of every instruction and *before* fetching the following one;
  - Upon a notification, a procedure is called at a fixed location: *“interrupt handler”*
  - Upon completion of the interrupt handler, the following instruction is executed (as if nothing happened)
- The interrupt handler takes care of the I/O.

Interrupts: Remarks

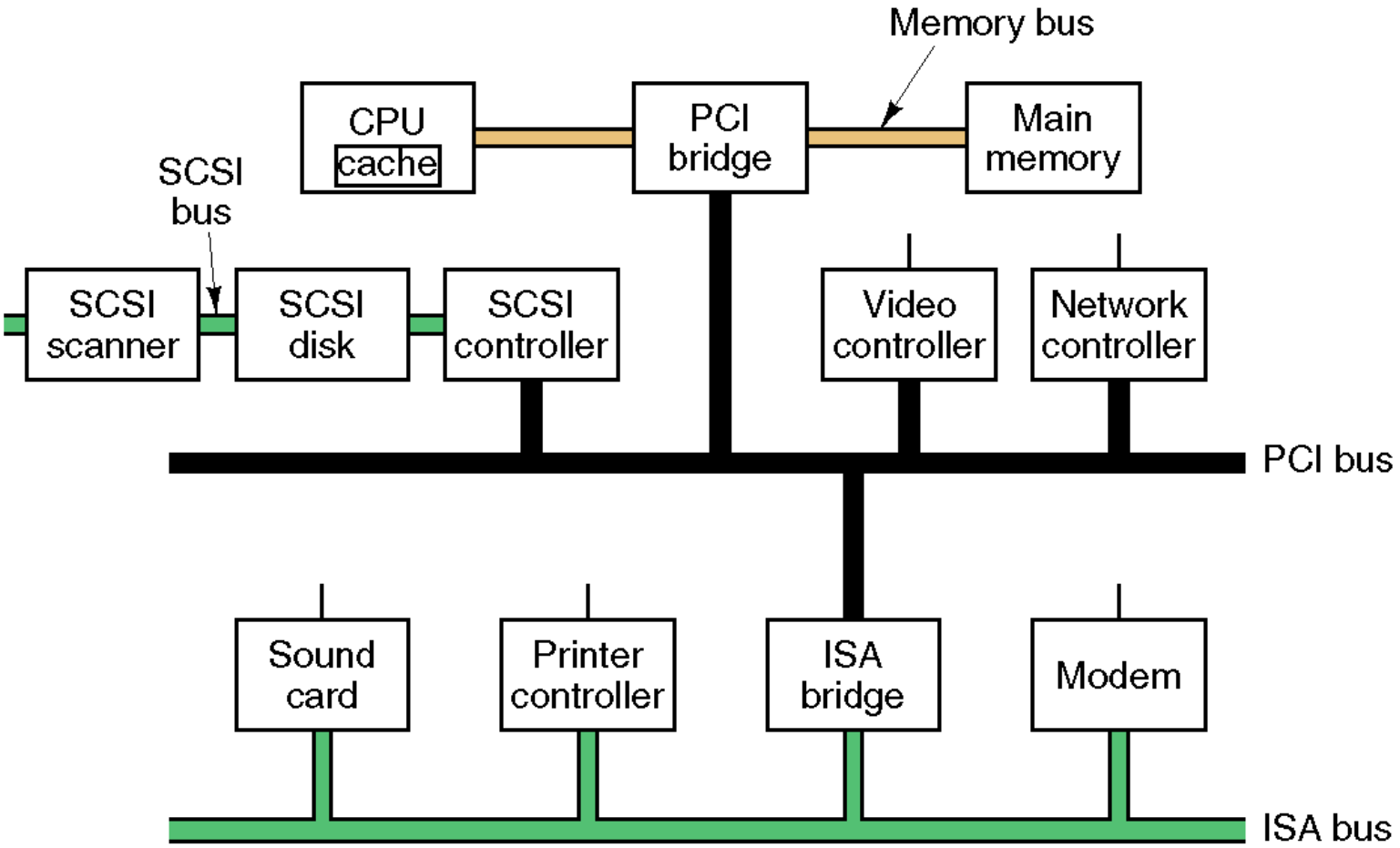
- Interrupt handlers must be **transparent**, i.e.
  1. the state of the interrupted process is saved;
  2. the interrupt handled;
  3. the state of the interrupted process is restored to exactly the same condition it was in when the interrupt occurred;
  4. the process is resumed
- Whenever an interrupt takes too long (e.g. longer than the period of the interrupt) the main program can't do its work any more.
- For time-critical interrupts (e.g. burning a CD) **high-priority** interrupts are used.

Interrupts: Remarks

A microprocessor typically provides:

- different types of interrupts, e.g.
  - System reset;
  - (Non)-maskable NMI, which are not masked by the general interrupt enable bit, but by individual interrupt enable bits;
  - Maskable, where each interrupt source can typically also be disabled individually by an interrupt enable bit next to the ability to disable all maskable interrupts by a general interrupt enable bit (e.g. in the status register).
- multiple interrupt levels and nesting of interrupts.

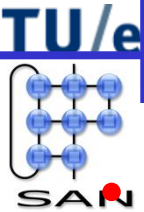
A modern PC: a further subdivision



Contents

- Computer Systems Organization – basics
- Parallelism and cache
  - parallelism at instruction and processor level
  - cache memories
- Further Reading

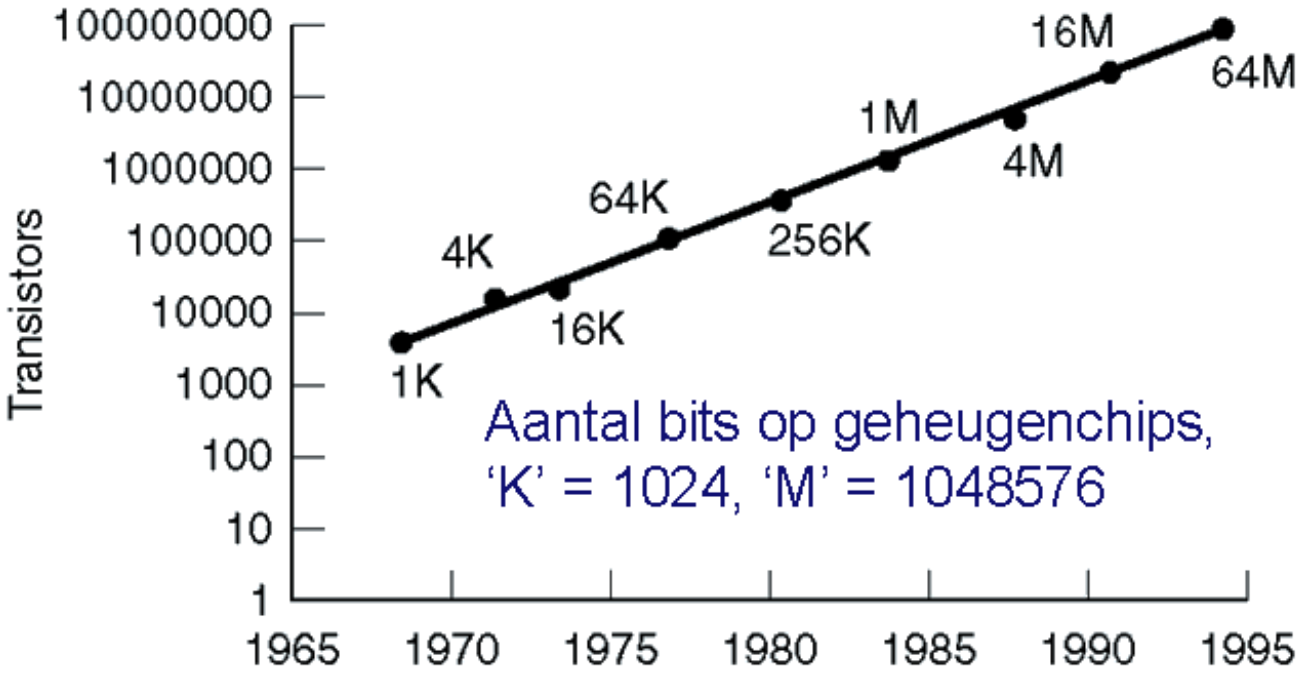
Parallelism and cache



- Computers become ever faster:
  - Moore's Law's (next slide)
  - Parallelism
    - at the level of instructions;
    - at the level of processors.

"Moore's Law"

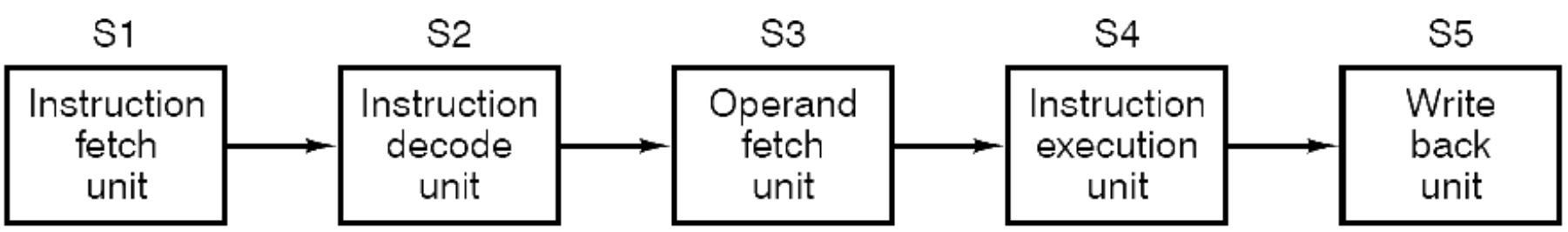
- Prediction in 1965 by the founder of Intel:
  "number of transistors per chip doubles every 18 month" (60% growth per year)



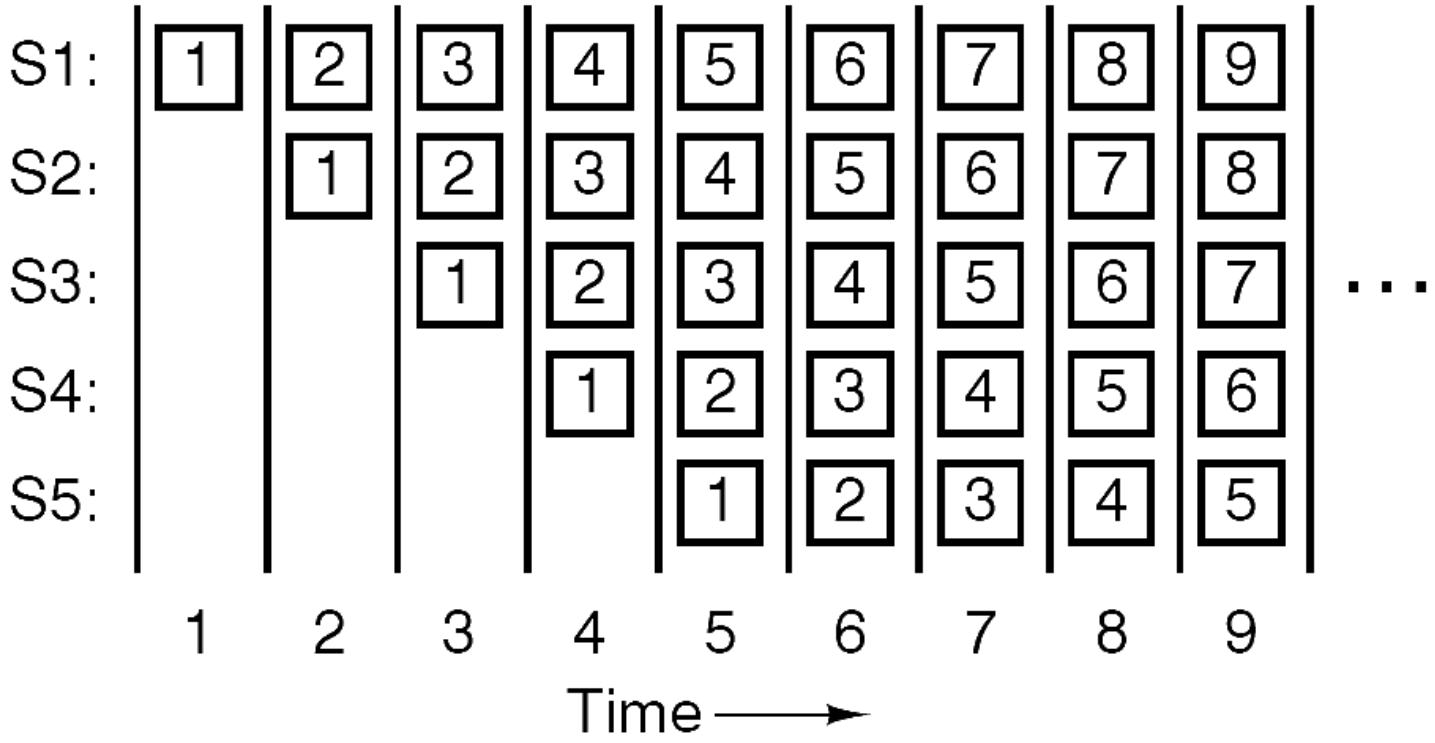
Gordon E. Moore, *Cramming more components onto circuits*, Electronics, 38(8), April 1965

Parallelism at the instruction-level

- Prefetch buffer:
  - *Problem*: the memory is too slow
  - *Solution*: a prefetch buffer to store next instructions
  - Already applied by IBM' Stretch processor in 1959!
- Extending the idea: Pipelining:



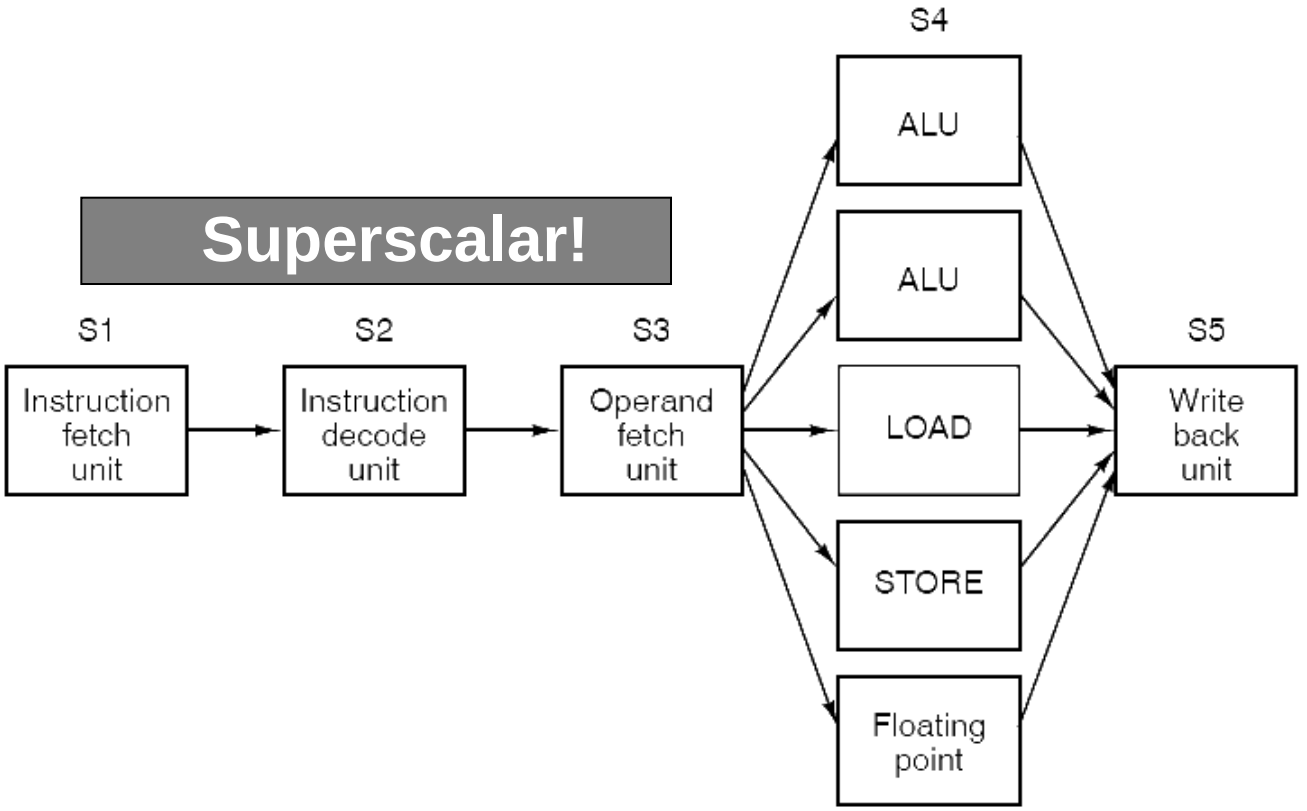
Pipeline timing:



Pipeline:

- Cycletime *T* and *n* modules:
  - Latency (duration of an instruction): *nT*
  - Bandwidth: *1/T* MIPS
- **Intermezzo:** Clockspeed is a bad metric:
  - MIPS (Millions Instructions Per Second) is better
  - MFLOPS is very convenient for calculations (Millions FLoating point OPeration per Second)
  - These are *computer*-metrics rather than *processor*-metrics.

A pipeline with multiple units (VLIW)

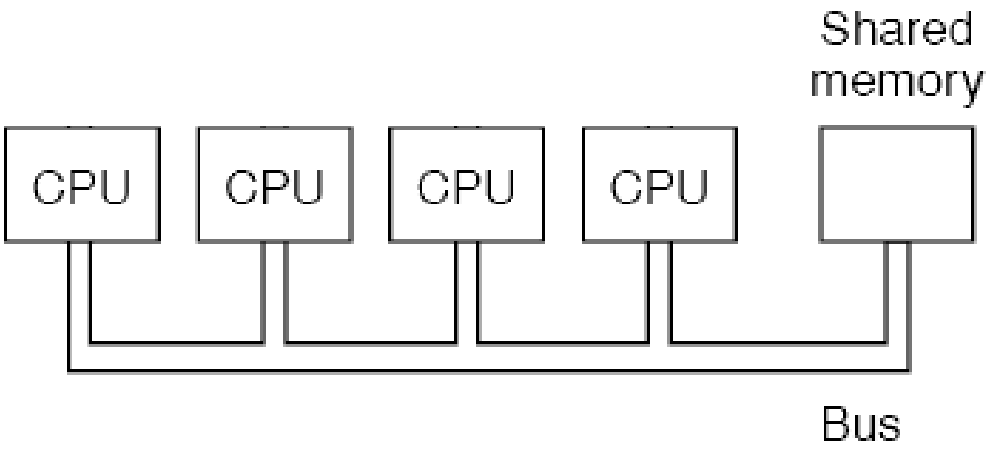


Note: S4 is (much) slower than S1, S2 and S3

Parallelism at the processor-level

- Pipelining and superscalar: 5 to 10 x faster
- That's (of course…) not enough
- For a factor 50 to 100, we use multiple CPUs, e.g. multiprocessors
  - multiple independent CPUs;
  - shared memory using a shared bus;
  - for performance reasons, each CPU has typically some local memory.

Multiprocessor system

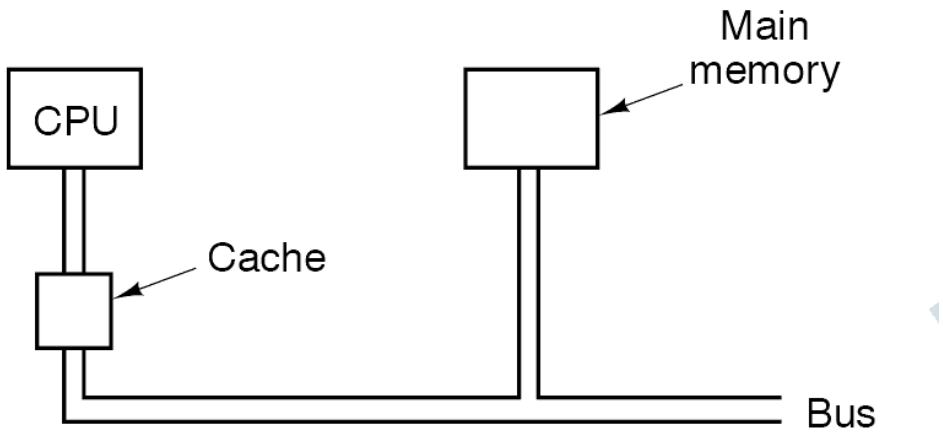


Cache memories

- *Problem*: memory is too slow
- *Solution*: memories become faster
- *Problem*: CPUs as well (become much faster)
- Note:
  - Hence, memories become relatively slower
  - Fast memory exists, but is expensive
- *Idea*: try to use as less fast memory as possible to improve the overall speed as much as possible

Cache memories

- Based on the “locality principle”:
  - A program seldomly performs Random Memory Acces, i.e. next references (instructions as well as data) are typically close to previous references
  - A cache is a fast auxiliary memory between main memory and the CPU



Cache memories

Before a Memory Access, first look in the cache.

If it is not available, use main memory.

Fetch multiple consecutive addresses from memory, rather than one, and store the information in the cache.

Main memory has been designed to support this kind of access

Notes:

- caches are typically invisible to a programme;
- there may exist multiple levels of caches;
- there may be different types of caches (data and instruction).

Further reading

- [Tanenbaum 06] Andrew S. Tanenbaum, *Structured Computer Organization*, 5th edition, Pearson Prentice Hall, 2006.